Logical Induction (Abridged)

Scott Garrabrant, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and **Jessica Taylor**
{scott,tsvi,critch,nate,jessica}@intelligence.org
Machine Intelligence Research Institute

Abstract

We describe a *logical induction criterion* for algorithms that reason probabilistically about logical facts, such as claims about mathematical conjectures and long-running computer programs. We show that algorithms which meet this criterion (*logical inductors*) satisfy a number of desirable properties, including (1) they develop accurate beliefs in a manner that outpaces deduction, by learning to assign high probabilities to provable conjectures and low probabilities to disprovable ones, long before the proofs can be found; (2) they learn to use appropriate statistical summaries to predict sequences of statements whose truth values appear pseudorandom; (3) their beliefs are coherent in the limit; (4) their beliefs strictly dominate the universal semimeasure in the limit; and (5) they learn to have accurate beliefs about their own current beliefs, in a manner that avoids the standard paradoxes of self-reference.

Roughly speaking, the logical induction criterion says that when a reasoner's beliefs are interpreted as prices in a stock market, it should not be possible to exploit that market via any strategy that can be expressed by a polynomial-time machine. This criterion bears strong resemblance to the "no Dutch book" criteria that support both expected utility theory (von Neumann and Morgenstern 1944) and Bayesian probability theory (Ramsey 1931; de Finetti 1937), and we think it captures a portion of what it means to do good reasoning under logical uncertainty.

This is an abridged version of a longer paper. In the extended paper, we give a computable *logical induction algorithm* that satisfies our criterion, and give proofs of the theorems stated here (along with many others).

1 Introduction

Every student of mathematics has experienced uncertainty about conjectures for which there is "quite a bit of evidence", such as the Riemann hypothesis or the twin prime conjecture. Indeed, when Zhang (2014) proved a bound on the gap between primes, we were tempted to increase our credence in the twin prime conjecture. But how much evidence does this bound provide for the twin prime conjecture? Can we quantify the degree to which it should increase our confidence?

The natural impulse is to appeal to probability theory in general and Bayes' theorem in particular. Bayes' theorem gives rules for how to use observations to update empirical uncertainty about unknown events in the physical world. However, probability theory lacks the tools to manage uncertainty about logical facts.

Consider encountering a computer connected to an input wire and an output wire. If we know what algorithm the computer implements, then there are two distinct ways to be uncertain about the output. We could be uncertain about the input—maybe it's determined by a coin toss we didn't see. Alternatively, we could

This is an abridged version of a longer paper (Garrabrant et al. 2016), which is available at https://intelligence.org/files/LogicalInduction.pdf. Published as Technical report 2016-2.

be uncertain because we haven't had the time to reason out what the program does—perhaps it computes the parity of the 87,653rd digit in the decimal expansion of π, and we don't personally know whether it's even or odd.

The first type of uncertainty is about *empirical* facts. No amount of thinking in isolation will tell us whether the coin came up heads. To resolve empirical uncertainty we must observe the coin, and then Bayes' theorem gives a principled account of how to update our beliefs.

The second type of uncertainty is about a *logical* fact, about what a known computation will output when evaluated. In this case, reasoning in isolation can and should change our beliefs: we can reduce our uncertainty by thinking more about π, without making any new observations of the external world.

In any given practical scenario, reasoners usually experience a mix of both empirical uncertainty (about how the world is) and logical uncertainty (about what that implies). In this paper, we focus entirely on the problem of managing logical uncertainty. Probability theory does not address this problem, because probability-theoretic reasoners cannot possess uncertainty about logical facts. For example, let ϕ stand for the claim that the 87,653rd digit of π is a 7. If this claim is true, then $(1 + 1 = 2) \Rightarrow \phi$. But the laws of probability theory say that if $A \Rightarrow B$ then $\Pr(A) \leq \Pr(B)$. Thus, a perfect Bayesian must be at least as sure of ϕ as they are that $1 + 1 = 2$! Recognition of this problem dates at least back to Good (1950).

Many have proposed methods for relaxing the criterion $\Pr(A) \leq \Pr(B)$ until such a time as the implication has been proven (see, e.g, the work of Hacking [1967] and Christiano [2014]). But this leaves open the question of how probabilities should be assigned before the implication is proven, and this brings us back to the search for a principled method for managing uncertainty about logical facts when relationships between them are suspected but unproven.

We propose a partial solution, which we call *logical induction*. Very roughly, our setup works as follows. We consider reasoners that assign probabilities to sentences written in some formal language and refine those probabilities over time. Assuming the language is sufficiently expressive, these sentences can say things like "Goldbach's conjecture is true" or "the computation `prg` on input `i` produces the output `prg(i)=0`". The reasoner is given access to a slow deductive process that emits theorems over time, and tasked with assigning probabilities in a manner that outpaces deduction, e.g., by assigning high probabilities to sentences that are eventually proven, and low probabilities to sentences that are eventually refuted, well before they can be verified deductively. Logical inductors carry out this task in a way that satisfies many desirable properties, including:

1. Their beliefs are logically consistent in the limit as time approaches infinity.
2. They learn to make their probabilities respect many different patterns in logic, at a rate that outpaces deduction.
3. They learn to know what they know, and trust their future beliefs, while avoiding paradoxes of self-reference.

These claims (and many others) will be made precise in Section 4.

A logical inductor is any sequence of probabilities that satisfies our *logical induction criterion*, which works roughly as follows. We interpret a reasoner's probabilities as prices in a stock market, where the probability of ϕ is interpreted as the price of a share that is worth \$1 if ϕ is true, and \$0 otherwise (similar to Beygelzimer, Langford, and Pennock [2012]). We consider a collection of stock traders who buy and sell shares at the market prices, and define a sense in which traders can exploit markets that have irrational beliefs. The logical induction criterion then says that it should not be possible to exploit the market prices using any trading strategy that can be generated in polynomial-time.

Our main finding is a computable algorithm which satisfies the logical induction criterion, and hence exhibits the properties listed above.

The logical induction criterion can be seen as a weakening of the "no Dutch book" criterion that Ramsey (1931) and de Finetti (1937) used to support standard probability theory, which is analogous to the "no Dutch book" criterion that von Neumann and Morgenstern (1944) used to support expected utility theory. Because

of the analogy, and the variety of desirable properties that follow immediately from this one criterion, we believe that the logical induction criterion captures a portion of what it means to do good reasoning about logical facts in the face of deductive limitations. That said, there are clear drawbacks to our algorithm: it does not use its resources efficiently; it is not a decision-making algorithm (i.e., it does not "think about what to think about"); and the properties above hold either asymptotically (with poor convergence bounds) or in the limit. In other words, our algorithm gives a theoretically interesting but ultimately impractical account of how to manage logical uncertainty.

1.1 Note on abridgment

This paper is an abridged version of a longer paper. In this paper, we describe the logical induction framework and state a number of properties that follow from it. In the extended version, we give the logical induction algorithm; proofs that the properties stated here follow from the logical induction criterion; a fuller account of the related work; a discussion of properties not listed here (calibration, conditional probabilities, etc.); and a more in-depth discussion of our framework and its applications. The extended paper can be found at `https://intelligence.org/files/LogicalInduction.pdf` (Garrabrant et al. 2016).

1.2 Related work

The problem of reasoning under logical uncertainty has been studied for quite some time, and in this abridgement, we lack the space to give anything except the barest of treatments. Study of the problem dates back to Bernoulli and Boole (1854); the problem was laid out in a particularly clear fashion by Good (1950). We take the approach of Gaifman (1964), which is also taken by Demski (2012) and Hutter et al. (2013). Our framework is heavily inspired by that of Solomonoff (1964) and later work on that framework done by Zvonkin and Levin (1970) and Li and Vitányi (1993). The related work in the field of AI is extensive; see Potyka and Thimm (2015), Richardson and Domingos (2006), and Briol et al. (2015) in particular. Our interpretation of probabilities as prices in a stock market is quite similar to that of Beygelzimer, Langford, and Pennock (2012), and our philosophical stance is quite similar in spirit to the one advocated by Aaronson (2013). This summary is woefully incomplete; refer to the extended version for a fuller account.

1.3 Overview

In Section 2 we define some notation. In Section 3 we state the logical induction criterion and our main theorem. The logical induction criterion is motivated by a series of stock trading analogies, which are also introduced in 3. In Section 4 we discuss a number of properties that follow from the logical induction criterion, including limit properties (4.1); pattern recognition properties (4.2); and (3) properties of self-knowledge and self-trust (4.3). For the algorithm, proofs that these properties and others follow from the criterion, and a discussion of the framework and its applications, see the extended paper.

2 Notation

This section defines notation used throughout the paper. The reader is invited to skim it, or perhaps skip it entirely and use it only as a reference when needed.

We denote the set of positive natural numbers by $\mathbb{N}^+$, where the superscript makes it clear that 0 is not included. We work with $\mathbb{N}^+$ instead of $\mathbb{N}^{\geq 0}$ because we regularly consider initial segments of infinite sequences up to and including the element at index n, and it will be convenient for those lists to have length n. When considering continuous functions with range in $\mathbb{Q}$, we use the subspace topology on $\mathbb{Q}$ inherited from $\mathbb{R}$. We use $\mathbb{B}$ to denote the set $\{0, 1\}$ interpreted as Boolean values.

We generally use the symbols ϕ, ψ, χ to denote well-formed formulas in some language of propositional logic $\mathcal{L}$ (such as a theory of first order logic, where the propositional atoms are prime formulas), which includes the basic logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and uses modus ponens as its rule of inference. We assume that $\mathcal{L}$ has been chosen so that its sentences can be interpreted as claims about some class of mathematical objects, such as natural numbers or computer programs.

We will write logical formulas inside quotes "$-$", such as $\phi :=$ "$x = 3$". We sometimes write things like $\phi :=$ "Goldbach's conjecture", in which case it is understood that the English text could be expanded into a precise arithmetical claim. We use underlines to indicate when a symbol in a formula should be replaced by the expression it stands for. For example, if $n := 3$, then $\phi :=$ "$x > \underline{n}$" means $\phi =$ "$x > 3$", and $\psi :=$ "$\underline{\phi} \rightarrow (x = \underline{n} + 1)$" means $\psi =$ "$x > 3 \rightarrow (x = 3 + 1)$". If ϕ and ψ denote formulas, then $\neg\phi$ denotes "$\neg(\underline{\phi})$" and $\phi \wedge \psi$ denotes "$(\underline{\phi}) \wedge (\underline{\psi})$" and so on. For instance, if $\phi :=$ "$x > 3$" then $\neg\phi$ denotes "$\neg(x > 3)$".

When we say a theory Γ "can represent computable functions", we mean that its language is used to refer to computer programs in such a way that Γ satisfies the representability theorem for computable functions, which means that for every (total) computable function $f : \mathbb{N}^+ \rightarrow \mathbb{N}^+$, there exists a Γ-formula γ_f with two free variables such that for all $n \in \mathbb{N}^+$, Γ proves "$\forall \nu \colon \gamma_f(\underline{n}, \nu) \leftrightarrow \nu = \underline{f(n)}$", where "$\gamma_f(\underline{n}, \nu)$" stands, in the usual way, for the formula resulting from substituting an encoding of n and the symbol ν for its free variables. In this case, we use "$\underline{f}(\underline{n})$" as shorthand for the formula "$\gamma_f(\underline{n}, \nu)$". By convention, we use "$\underline{f}(3) < \underline{g}(3)$" as shorthand for

$$\text{“}\forall xy \colon \gamma_f(3, x) \wedge \gamma_g(3, y) \rightarrow x < y\text{”},$$

and so on. In particular, note that writing down a sentence like "$\underline{f}(3) > 4$" does not involve computing the value $f(3)$; it merely requires writing out the definition of γ_f. This distinction is important when f has a very slow runtime.

We denote infinite sequences using overlines, like $\overline{x} := (x_1, x_2, \ldots)$, where it is understood that x_i denotes the ith element of $\overline{x}$, for $i \in \mathbb{N}^+$. To define sequences of sentences compactly, we use parenthetical expressions such as $\overline{\phi} :=$ ("$\underline{n} > 7$")$_{n \in \mathbb{N}^+}$. We define $x_{\leq n} := (x_1, \ldots, x_n)$. Given another element y, we abuse notation in a common way and define $(x_{\leq n}, y) = (x_1, \ldots, x_n, y)$ to be the list $x_{\leq n}$ with y appended at the end. We write $()$ for the empty sequence.

Given any sequences $\overline{x}$ and $\overline{y}$, we write

$$\begin{aligned}
x_n \eqsim_n y_n \quad &\text{for} \quad \lim_{n\to\infty} x_n - y_n = 0, \\
x_n \gtrsim_n y_n \quad &\text{for} \quad \liminf_{n\to\infty} x_n - y_n \geq 0, \text{ and} \\
x_n \lesssim_n y_n \quad &\text{for} \quad \limsup_{n\to\infty} x_n - y_n \leq 0.
\end{aligned}$$

3 The logical induction criterion

In this section, we will develop a framework in which we can state the logical induction criterion and the properties that follow from it. This will culminate in the following definition and theorem:

Definition 3.0.1 (The Logical Induction Criterion)**.** *A market $\overline{\mathbb{P}}$ is said to satisfy the* ***logical induction criterion*** *relative to a deductive process $\overline{D}$ if there is no efficiently computable trader $\overline{T}$ that exploits $\overline{\mathbb{P}}$ relative to $\overline{D}$. A market $\overline{\mathbb{P}}$ meeting this criterion is called a* ***logical inductor over $\overline{D}$****.*

Theorem 3.0.2. *For any deductive process $\overline{D}$, there exists a computable belief sequence $\overline{\mathbb{P}}$ satisfying the logical induction criterion relative to $\overline{D}$.*

We will now define markets, deductive processes, efficient computability, traders, and exploitation.

3.1 Markets

We will be concerned with methods for assigning values in the interval $[0, 1]$ to sentences of logic. We will variously interpret those values as prices, probabilities, and truth values, depending on the context. Let $\mathcal{L}$ be a language of propositional logic, and let $\mathcal{S}$ be the set of all sentences written in $\mathcal{L}$. We then define:

Definition 3.1.1 (Valuation)**.** *A **valuation** is any function $\mathbb{V} : \mathcal{S} \to [0, 1]$. We refer to $\mathbb{V}(\phi)$ as the value of ϕ according to $\mathbb{V}$. A valuation is called rational if its image is in $\mathbb{Q}$.*

First let us treat the case where we interpret the values as prices.

Definition 3.1.2 (Pricing)**.** *A **pricing** $\mathbb{P} : \mathcal{S} \to \mathbb{Q} \cap [0, 1]$ is any computable rational valuation. If $\mathbb{P}(\phi) = p$ we say that the price of a ϕ-share according to $\mathbb{P}$ is p, where the intended interpretation is that a ϕ-share is worth \$1 if ϕ is true.*

Definition 3.1.3 (Market)**.** *A **market** $\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, \ldots)$ is a computable sequence of pricings $\mathbb{P}_i : \mathcal{S} \to \mathbb{Q} \cap [0, 1]$.*

We can visualize a market as a series of pricings that may change day by day. The properties proven in Section 4 will apply to any market that satisfies the logical induction criterion. Theorem 4.1.2 (Limit Coherence) will show that the prices of a logical inductor can reasonably be interpreted as probabilities, so we will often speak as if the prices in a market represent the beliefs of a reasoner, where $\mathbb{P}_n(\phi) = 0.75$ is interpreted as saying that on day n, the reasoner assigns 75% probability to ϕ.

In fact, the logical inductor that we construct in the extended paper has the additional property of being finite at every timestep, which means we can visualize it as a series of finite belief states that a reasoner of interest writes down each day.

Definition 3.1.4 (Belief State)**.** *A **belief state** $\mathbb{P} : \mathcal{S} \to \mathbb{Q} \cap [0, 1]$ is a computable rational valuation with finite support, where $\mathbb{P}(\phi)$ is interpreted as the probability of ϕ (which is 0 for all but finitely many ϕ).*

We can visualize a belief state as a finite list of (ϕ, p) pairs, where the ϕ are unique sentences and the p are rational-number probabilities, and $\mathbb{P}(\phi)$ is defined to be p if (ϕ, p) occurs in the list, and 0 otherwise.

Definition 3.1.5 (Computable Belief Sequence)**.** *A **computable belief sequence** $\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, \ldots)$ is a computable sequence of belief states, interpreted as a reasoner's explicit beliefs about logic as they are refined over time.*

We can visualize a computable belief sequence as a large spreadsheet where each column is a belief state, and the rows are labeled by an enumeration of all logical sentences. We can then imagine a reasoner of interest working on this spreadsheet, by working on one column per day.

Philosophically, the reason for this setup is as follows. Most people know that the sentence "$1 + 1$ is even" is true, and that the sentence "$1 + 1 + 1 + 1$ is even" is true. But consider, is the following sentence true?

$$\text{“}1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 \text{ is even”}$$

To answer, we must pause and count the ones. Since we wish to separate the question of what a reasoner already knows from what they could infer using further computing resources, we require that the reasoner write out their beliefs about logic explicitly, and refine them day by day.

In this framework, we can visualize a reasoner as a person who computes the belief sequence by filling in a large spreadsheet, always working on the nth column on the nth day, by refining and extending her previous work as she learns new facts and takes more sentences into account, while perhaps making use of computer

assistance. For example, a reasoner who has noticed that "$1 + \cdots + 1$ is even" is true iff the sentence has an even number of ones, might program her computer to write 1 into as many of the true "$1 + \cdots + 1$ is even" cells per day as it can before resources run out. As another example, a reasoner who finds a bound on the prime gap might go back and update her probability on the twin prime conjecture. In our algorithm, the reasoner will have more and more computing power each day, with which to construct her next belief state.

3.2 Deductive processes

We are interested in the question of what it means for reasoners to assign "reasonable probabilities" to statements of logic. Roughly speaking, we will imagine reasoners that have access to some formal deductive process, such as a community of mathematicians who submit machine-checked proofs to an official curated database. We will study reasoners that "outpace" this deductive process, e.g., by assigning high probabilities to conjectures that will eventually be proven, and low probabilities to conjectures that will eventually be disproven, well before the relevant proofs are actually found.

Definition 3.2.1 (Deductive process)**.** *A **deductive process** $\overline{D} : \mathbb{N}^+ \to \mathrm{Fin}(\mathcal{S})$ is a computable nested sequence $D_1 \subseteq D_2 \subseteq D_3 \ldots$ of finite sets of sentences. We write D_∞ for the union $\bigcup_n D_n$.*

This is a rather barren notion of "deduction". We will consider cases where we fix some theory Γ, and D_n is interpreted as the theorems proven up to and including day n. In this case, $\overline{D}$ can be visualized as a slow process that reveals the knowledge of Γ over time. Roughly speaking, we will mainly concern ourselves with the case where $\overline{D}$ eventually rules out all and only the worlds that are inconsistent with Γ.

Definition 3.2.2 (World)**.** *A **world** is any truth assignment $\mathbb{W} : \mathcal{S} \to \mathbb{B}$. If $\mathbb{W}(\phi) = 1$ we say that ϕ is **true in** $\mathbb{W}$. If $\mathbb{W}(\phi) = 0$ we say that ϕ is **false in** $\mathbb{W}$. We write $\mathcal{W}$ for the set of all worlds.*

Observe that worlds are valuations, and that they are not necessarily consistent. This terminology is nonstandard; the term "world" is usually reserved for consistent truth assignments. Logically uncertain reasoners cannot immediately tell which truth assignments are inconsistent, because revealing inconsistencies requires time and effort. In fact, there are multiple notions of "consistency" that are relevant for our purposes:

Definition 3.2.3 (Propositional Consistency)**.** *A world $\mathbb{W}$ is called **propositionally consistent**, abbreviated **p.c.**, if for all $\phi \in \mathcal{S}$, $\mathbb{W}(\phi)$ is determined by Boolean algebra from the truth values that $\mathbb{W}$ assigns to the prime formulas of ϕ. In other words, $\mathbb{W}$ is p.c. if $\mathbb{W}(\phi \wedge \psi) = \mathbb{W}(\phi) \wedge \mathbb{W}(\psi)$, $\mathbb{W}(\phi \vee \psi) = \mathbb{W}(\phi) \vee \mathbb{W}(\psi)$, and so on.*

*Given a set of sentences D, we define $\mathcal{PC}(D)$ to be the set of all p.c. worlds where $\mathbb{W}(\phi) = 1$ for all $\phi \in D$. We refer to $\mathcal{PC}(D)$ as the set of worlds **propositionally consistent with D**.*

Definition 3.2.4 (Consistency)**.** *Given a set of sentences Γ, we say $\mathbb{W}$ is **consistent with Γ** or **Γ-consistent** if*

$$\Gamma \cup \{\phi \mid \mathbb{W}(\phi) = 1\} \cup \{\neg\phi \mid \mathbb{W}(\phi) = 0\} \nvdash \bot.$$

We write $\mathcal{C}(\Gamma)$ for the set of Γ-consistent worlds.

Note that a limited reasoner won't be able to tell whether a given world $\mathbb{W}$ is consistent (by either definition). Roughly speaking, the definition of exploitation (below) will say that a good reasoner should perform well when measured on day n by worlds propositionally consistent with D_n, and we ourselves will be interested in deductive processes that pin down a particular theory Γ by propositional consistency:

Definition 3.2.5 (Γ-complete)**.** *Given a theory Γ, we say that a deductive process $\overline{D}$ is* ***Γ-complete*** *if*

$$\mathcal{PC}(D_\infty) = \mathcal{C}(\Gamma).$$

As a canonical example, let D_n be the set of all theorems of PA provable in at most n characters.[1] Then $\overline{D}$ is PA-complete, and a reasoner with access to $\overline{D}$ can be interpreted as someone who on day n knows all PA-theorems provable in $\leq n$ characters, who must manage her uncertainty about other mathematical facts.

3.3 Efficient computability

We use the following notion of efficiency throughout the paper:

Definition 3.3.1 (Efficiently computable)**.** *An infinite sequence $\overline{x}$ is called* ***efficiently computable****, abbreviated* ***e.c.****, if there is a computable function f that outputs x_n on input n, with runtime polynomial in n.*

Our framework is not wedded to this definition; stricter notions of efficiency (e.g., sequences that can be computed in $\mathcal{O}(n^2)$ time) would yield "dumber" inductors with better runtimes, and vice versa. We use the set of polynomial-time computable functions because it has some closure properties that are convenient for our purposes.

3.4 Traders

Roughly speaking, traders are functions that see the day n and the history of market prices up to and including day n, and then produce a series of buy and sell orders, by executing a strategy that is continuous as a function of the market history.

A linear combination of sentences can be interpreted as a "market order", where $3\phi - 2\psi$ says to buy 3 shares of ϕ and sell 2 shares of ψ. Very roughly, a trading strategy for day n will be a method for producing market orders where the coefficients are not numbers but *functions* which depend (continuously) on the market prices up to and including day n.

Definition 3.4.1 (Valuation Feature)**.** *A valuation* ***feature*** $\alpha : [0,1]^{\mathcal{S}\times\mathbb{N}^+} \to \mathbb{R}$ *is a continuous function from valuation sequences to real numbers such that $\alpha(\overline{\mathbb{V}})$ depends only on the initial sequence $\mathbb{V}_{\leq n}$ for some $n \in \mathbb{N}^+$ called the* rank *of the feature,* $\mathrm{rank}(\alpha)$*. For any $m \geq n$, we define $\alpha(\mathbb{V}_{\leq m})$ in the natural way. We will often deal with features that have range in $[0,1]$; we call these $[0,1]$-features.*

The following valuation features find the price of a sentence on a particular day:

Definition 3.4.2 (Price Feature)**.** *For each $\phi \in \mathcal{S}$ and $n \in \mathbb{N}^+$, we define a* ***price feature*** $\phi^{*n} : [0,1]^{\mathcal{S}\times\mathbb{N}^+} \to \mathbb{R}$ *by the formula*

$$\phi^{*n}(\overline{\mathbb{V}}) := \mathbb{V}_n(\phi).$$

We call these "price features" because they will almost always be applied to a market $\overline{\mathbb{P}}$, in which case ϕ^{*n} gives the price of ϕ on day n as a function of $\overline{\mathbb{P}}$.

Very roughly, trading strategies will be linear combinations of sentences where the coefficients are valuation features. The set of all valuation features is not computably enumerable, so we define an expressible subset:

Definition 3.4.3 (Expressible Feature)**.** *An* ***expressible feature*** $\xi : [0,1]^{\mathcal{S}\times\mathbb{N}^+} \to \mathbb{R}$ *is a valuation feature expressible by an algebraic expression built from price features ϕ^{*n} for each $n \in \mathbb{N}^+$ and $\phi \in \mathcal{S}$, rational numbers, addition, multiplication,*

1. Because PA is a first-order theory, and the only assumption we made about $\mathcal{L}$ is that it is a propositional logic, note that the axioms of first-order logic—namely, specialization and distribution—must be included as theorems in $\overline{D}$.

$\max(-,-)$, *and a "safe reciprocation" function* $\max(-,1)^{-1}$. *See the extended paper for the full definition.*[2]

We write $\mathcal{EF}$ *for the set of all expressible features and* $\mathcal{EF}_n$ *for the set of expressible features of rank* $\leq n$.

Note that for each n, $\mathcal{EF}_n$ is a commutative ring. We will write $2 - \phi^{*6}$ for the function $\overline{\mathbb{V}} \mapsto 2 - \phi^{*6}(\overline{\mathbb{V}})$ and so on, in the usual way. For example, the feature

$$\xi := \max(0, \phi^{*6} - \psi^{*7})$$

checks whether the value of ϕ on day 6 is higher than the value of ψ on day 7. If so, it returns the difference; otherwise, it returns 0. If ξ is applied to a market $\overline{\mathbb{P}}$, and $\mathbb{P}_6(\phi) = 0.5$ and $\mathbb{P}_7(\psi) = 0.2$ then $\xi(\overline{\mathbb{P}}) = 0.3$. Observe that $\mathrm{rank}(\xi) = 7$, and observe that ξ is continuous.

The reason for the continuity constraint is to allow stable market prices to be found in the face of paradoxes. For example, consider the sentence ϕ which says "the price of ϕ in this market on day n is less than 50 cents" (constructed, e.g., by Gödel's diagonal lemma). What should the fair price be? A share of ϕ is worth \$1 if its price is less than 50 cents, and \$0 otherwise. Thus, if the price of ϕ is low, it is worth a lot; but if the price is high, it is worthless. If traders are allowed to buy ϕ on day n if it costs less than 50 cents and sell otherwise, then there are no stable market prices. The continuity constraint requires that if a trader buys at one price and sells at another, there must be an intermediate price at which they buy/sell nothing. As we will see later, this guarantees that a stable fixed point can be found between market prices and trading strategies. Intuitively, the continuity condition can be interpreted as saying that traders have only finite-precision access to the market prices.

We are almost ready to define trading strategies as a linear combination of sentences with expressible features as coefficients. However, there is one more complication. It will be convenient to record not only the amount of shares bought and sold, but amount of cash spent or received. For example, consider again the market order $3\phi - 2\psi$. If it is executed on day 7 in a market $\overline{\mathbb{P}}$, and $\mathbb{P}_7(\phi) = 0.4$ and $\mathbb{P}_7(\psi) = 0.3$, then the cost is $3 \cdot 40¢ - 2 \cdot 30¢ = 60¢$. We can record the whole trade as an affine combination $-0.6 + 3\phi - 2\psi$, which can be read as "the trader spent 60 cents to buy 3 shares of ϕ and sell 2 shares of ψ". Extending this idea to the case where the coefficients are expressible features, we get the following notion:

Definition 3.4.4 (Trading Strategy)**.** *A* ***trading strategy for day*** n*, also called an* ***n-strategy****, is an affine combination of the form*

$$T = c + \xi_1\phi_1 + \cdots + \xi_k\phi_k,$$

where $\phi_1, \ldots, \phi_k$ *are sentences,* $\xi_1, \ldots, \xi_k$ *are expressible features of rank* $\leq n$*, and*

$$c = -\sum_i \xi_i {\phi_i}^{*n}$$

is a "cash term" recording the net cash flow when executing a transaction that buys ξ_i *shares of* ϕ_i *for each* i *at the prevailing market price. (Buying negative shares is called "selling".) We define* $T[1]$ *to be* c*, and* $T[\phi]$ *to be the coefficient of* ϕ *in* T*, which is* 0 *if* $\phi \notin (\phi_1, \ldots, \phi_k)$.

An n*-strategy* T *can be encoded by the tuples* $(\phi_1, \ldots \phi_k)$ *and* $(\xi_1, \ldots \xi_k)$ *because the* c *term is determined by them. Explicitly, by linearity we have*

$$T = \xi_1 \cdot (\phi_1 - {\phi_1}^{*n}) + \cdots + \xi_k \cdot (\phi_k - {\phi_k}^{*n}),$$

which means any n*-strategy can be written as a linear combination of* $(\phi_i - {\phi_i}^{*n})$ *terms, each of which means "buy one share of* ϕ_i *at the prevailing price".*

2. In particular, expressible features are a generalization of arithmetic circuits. The specific definition is somewhat arbitrary; what matters is that expressible features be (1) continuous; (2) compactly specifiable in polynomial time; and (3) expressive enough to identify a variety of inefficiencies in a market.

As an example, consider the following trading strategy for day 5:

$$\left[(\neg\neg\phi)^{*5} - \phi^{*5}\right] \cdot \left(\phi - \phi^{*5}\right) + \left[\phi^{*5} - (\neg\neg\phi)^{*5}\right] \cdot \left(\neg\neg\phi - (\neg\neg\phi)^{*5}\right).$$

This strategy compares the price of ϕ on day 5 to the price of $\neg\neg\phi$ on day 5. If the former is less expensive by δ, it purchase δ shares of ϕ at the prevailing prices, and sells δ shares of $\neg\neg\phi$ at the prevailing prices. Otherwise, it does the opposite. In short, this strategy arbitrages ϕ against $\neg\neg\phi$ by buying the cheaper one and selling the more expensive one.

We can now state the key definition of this section:

Definition 3.4.5 (Trader)**.** *A **trader** $\overline{T}$ is a sequence $(T_1, T_2, \ldots)$ where each T_n is a trading strategy for day n.*

We can visualize a trader as a person who gets to see the day n, think for a while, and then produce a trading strategy for day n, which will observe the history of market prices up to and including day n and execute a market order to buy and sell different sentences at the prevailing market prices.

If $s := T_n[\phi] > 0$, we say that $\overline{T}$ buys s shares of ϕ on day n, and if $s < 0$, we say that $\overline{T}$ sells $|s|$ shares of ϕ on day n. Similarly, if $d := T_n[1] > 0$, we say that $\overline{T}$ receives d dollars on day n, and if $d < 0$, we say that $\overline{T}$ pays out $|d|$ dollars on day n.

Trade strategies are a special case of affine combinations of sentences:

Definition 3.4.6 (Affine Combination)**.** *An **$\mathcal{F}$-combination** $A : \mathcal{S} \cup \{1\} \to \mathcal{F}_n$ is an affine expression of the form*

$$A := c + \alpha_1\phi_1 + \cdots + \alpha_k\phi_k,$$

where $(\phi_1, \ldots, \phi_k)$ are sentences and $(c, \alpha_1, \ldots, \alpha_k)$ are in $\mathcal{F}$. $\mathcal{EF}$, $\mathbb{R}$, and $\mathbb{Q}$-combinations are defined similarly.

*We write $A[1]$ for the trailing coefficient c, and $A[\phi]$ for the coefficient of ϕ, which is 0 if $\phi \notin (\phi_1, \ldots, \phi_k)$. The **rank** of A is defined to be the maximum rank among all its coefficients. Given any valuation $\mathbb{V}$, we abuse notation in the usual way and define the **value** of A (according to $\mathbb{V}$) linearly by:*

$$\mathbb{V}(A) := c + \alpha_1\mathbb{V}(\phi_1) + \cdots + \alpha_k\mathbb{V}(\phi_k).$$

We will use these to encode the net holdings $\sum_{i \le n} T_i(\overline{\mathbb{P}})$ of a trader after interacting with a market $\overline{\mathbb{P}}$, and later to encode linear inequalities that hold between the truth values of different sentences.

3.5 Exploitation

Definition 3.5.1 (Exploitation)**.** *A trader $\overline{T}$ is said to **exploit** a valuation sequence $\overline{\mathbb{V}}$ relative to a deductive process $\overline{D}$ if the set of values*

$$\left\{\mathbb{W}\left(\textstyle\sum_{i \le n} T_i(\overline{\mathbb{V}})\right) \,\middle|\, n \in \mathbb{N}^+, \mathbb{W} \in \mathcal{PC}(D_n)\right\}$$

is bounded below, but not bounded above.

Given a world $\mathbb{W}$, the number $\mathbb{W}(\sum_{i \le n} T_i(\overline{\mathbb{P}}))$ is the value of the trader's net holdings, where a share of ϕ is valued at \$1 if ϕ is true in $\mathbb{W}$ and \$0 otherwise. The set $\{\mathbb{W}(\sum_{i \le n} T_i)(\overline{\mathbb{P}}) \mid n \in \mathbb{N}^+, \mathbb{W} \in \mathcal{PC}(D_n)\}$ is the set of all assessments of $\overline{T}$'s net worth, across all time, according to worlds that were propositionally consistent with $\overline{D}$ at the time. We informally call these *plausible assessments* of the trader's net worth. Using this terminology, Definition 3.5.1 says that a trader exploits the market if their plausible net worth is bounded below, but not above.

Roughly speaking, we can imagine that there is a person behind the market who acts as a market maker, obligated to buy and sell shares at the listed prices. We can imagine that anyone who sold a ϕ-share is obligated to pay \$1 if and when $\overline{D}$ says ϕ. Then, very roughly, a trader exploits the market if they are able to make unbounded returns off of a finite investment.

This analogy is illustrative but incomplete—traders can exploit the market even if they never purchase a sentence that appears in $\overline{D}$. For example, let ϕ and ψ be two sentences such that PA proves $(\phi \vee \psi)$, but does not prove either ϕ or ψ. Consider a trader that bought 10 ϕ-shares at a price of 20¢ each, and 10 ψ-shares at a price of 30¢ each. Once $\overline{D}$ says $(\phi \vee \psi)$, all remaining p.c. worlds will agree that the portfolio $-5 + 10\phi + 10\psi$ has a value of at least $+5$, despite the fact that neither ϕ nor ψ is ever proven. If the trader is allowed to keep buying ϕ and ψ shares at those prices, they would exploit the market, despite the fact that they never buy decidable sentences. In other words, our notion of exploitation rewards traders for arbitrage, even if they arbitrage between sentences that never "pay out".

3.6 Main Result

Recall the logical induction criterion and our main theorem:

Definition 3.0.1 (The Logical Induction Criterion)**.** *A market $\overline{\mathbb{P}}$ is said to satisfy the **logical induction criterion** relative to a deductive process $\overline{D}$ if there is no efficiently computable trader $\overline{T}$ that exploits $\overline{\mathbb{P}}$ relative to $\overline{D}$. A market $\overline{\mathbb{P}}$ meeting this criterion is called a **logical inductor over $\overline{D}$**.*

Theorem 3.0.2. *For any deductive process $\overline{D}$, there exists a computable belief sequence $\overline{\mathbb{P}}$ satisfying the logical induction criterion relative to $\overline{D}$.*

Proof. See the extended paper for the algorithm. □

Definition 3.6.1 (Logical Inductor over Γ)**.** *Given a theory Γ, a logical inductor over a Γ-complete deductive process $\overline{D}$ is called a **logical inductor over Γ**.*

Corollary 3.6.2. *For any recursively axiomatizable theory Γ, there exists a computable belief sequence that is a logical inductor over Γ.*

4 Properties of Logical Inductors

Here is an intuitive argument that logical inductors perform good reasoning under logical uncertainty:

> Consider any polynomial-time method for efficiently identifying patterns in logic. If the market prices don't learn to reflect that pattern, a clever trader can use it to exploit the market. For example, if there is a polynomial-time method for identifying theorems that are always underpriced by the market, a clever trader could use that pattern to buy those theorems low, exploiting the market. To avoid exploitation, logical inductors must learn to identify many different types of patterns in logical statements.

In this section, we will provide evidence supporting this intuitive argument, by demonstrating that a wide selection of desirable properties are satisfied in unison by logical inductors. In Section 4.1 we will discuss properties that hold in the limit; in Section 4.2 we will discuss properties related to pattern recognition; in Section 4.3 we will discuss properties related to self-knowledge and self-trust.

In what follows, let $\mathcal{L}$ be a language of propositional logic; let $\mathcal{S}$ be the set of sentences written in $\mathcal{L}$; let $\Gamma \subset \mathcal{S}$ be a computably enumerable set of propositional formulas written in $\mathcal{L}$ (such as PA, where the propositional variables are prime sentences in first-order logic, as discussed in Section 2); and let $\overline{\mathbb{P}}$ be a computable logical inductor over Γ, i.e., a market satisfying the logical induction criterion relative to some Γ-complete deductive process $\overline{D}$. We assume in this section that Γ is consistent.

4.1 Limit properties

The limiting beliefs of $\overline{\mathbb{P}}$ converge, and represent a coherent probability distribution:

Theorem 4.1.1 (Convergence)**.** *The limit* $\mathbb{P}_\infty : \mathcal{S} \to [0,1]$ *defined by*

$$\mathbb{P}_\infty(\phi) := \lim_{n\to\infty} \mathbb{P}_n(\phi)$$

exists for all ϕ*.* (Proof in extended paper, B.4.)

Theorem 4.1.2 (Limit Coherence)**.** $\mathbb{P}_\infty$ *is coherent, i.e., it gives rise to an internally consistent probability measure* Pr *on the set* $\mathcal{C}(\Gamma)$ *of all worlds consistent with* Γ*, defined by the formula*

$$\Pr(\mathbb{W}(\phi) = 1) := \mathbb{P}_\infty(\phi).$$

In other words, $\mathbb{P}_\infty$ *defines a probability measure on the set of completions of* Γ*.*

(Proof in extended paper, C.10.)

Furthermore, $\overline{\mathbb{P}}$ learns not to assign extreme probabilities to sentences that are independent from Γ. In fact, the limiting probability of an undecidable sentence is bounded away from 0 and 1 by an amount proportional to its prefix complexity:

Theorem 4.1.3 (Occam Bounds)**.** *There exists a fixed positive constant* C *such that for any sentence* ϕ *with prefix complexity* $\kappa(\phi)$*, if* $\Gamma \nvdash \neg\phi$*, then*

$$\mathbb{P}_\infty(\phi) \geq C2^{-\kappa(\phi)},$$

and if $\Gamma \nvdash \phi$*, then*

$$\mathbb{P}_\infty(\phi) \leq 1 - C2^{-\kappa(\phi)}.$$

(Proof in extended paper, G.3.)

By adding a sequence of zero-arity predicates $(b_1, b_2, \ldots)$ not mentioned in Γ to the language $\mathcal{L}$, $\mathbb{P}_\infty$ can be used as a full-fledged sequence predictor. Interpreting each b_i as representing the next bit in a sequence of bits produced by an environment, one can ask $\mathbb{P}_\infty$ the probability of a bitstring 00101 by asking for the probability of the sentence "$\neg b_1 \wedge \neg b_2 \wedge b_3 \wedge \neg b_4 \wedge b_5$". The answer will be proportional to the complexity of the bitstring, which means that $\mathbb{P}_\infty$ gives rise to a simplicity prior over bitstrings.

Theorem 4.1.4 (Domination of the Universal Semimeasure)**.** *Let* $(b_1, b_2, \ldots)$ *be a sequence of zero-arity predicate symbols in the language of* Γ *not mentioned in the axioms of* Γ*, and let* $\overline{\sigma}$ *be an infinite bitstring. Define*

$$\mathbb{P}_\infty(\overline{\sigma}_{\leq n}) = \mathbb{P}_\infty(\text{“}(b_1 \leftrightarrow \sigma_1 = 1) \wedge (b_2 \leftrightarrow \sigma_2 = 1) \wedge \ldots \wedge (b_n \leftrightarrow \sigma_n = 1)\text{”}).$$

Let M *be a universal semimeasure. Then there is some positive constant* C *such that for any finite bitstring* $\overline{\sigma}_{\leq n}$*,*

$$\mathbb{P}_\infty(\overline{\sigma}_{\leq n}) \geq C \cdot M(\overline{\sigma}_{\leq n}).$$

(Proof in extended paper, G.5.)

In the extended paper we show that this dominance is strict.

4.2 Pattern recognition

Theorem 4.2.1 (Provability Induction)**.** *Let* $\overline{\phi}$ *be an efficiently computable (e.c.) sequence of theorems. Then*

$$\mathbb{P}_n(\phi_n) \eqsim_n 1.$$

Furthermore, let $\overline{\psi}$ *be an e.c. sequence of disprovable sentences. Then*

$$\mathbb{P}_n(\psi_n) \eqsim_n 0.$$

(Proof in extended paper, C.3.)

To see why this theorem is interesting, consider an e.c. sequence $\overline{\phi}$ of theorems which can be generated in polynomial time, but are quite difficult to prove. Let $f(n)$ be the time at which $\overline{D}$ will prove ϕ_n, and assume that f is some fast-growing function. At any given time n, the statement ϕ_n is ever further out beyond the current deductive state D_n—it might take 1 day to prove ϕ_0, 10 days to prove ϕ_1, 100 days to prove ϕ_2, and so on. One might therefore expect that ϕ_n will also be "out of reach" for $\mathbb{P}_n$, and that we have to wait until a day close to $f(n)$ before expecting the prices $\mathbb{P}_{f(n)}(\phi_n)$ to be confident in ϕ_n. However, this is not the case!

Provability induction says that, for large n and a sequence $\overline{\phi}$ of theorems that can be efficiently enumerated, $\mathbb{P}_n(\phi_n) > 1 - \varepsilon$, despite the fact that ϕ_n will not be confirmed deductively until a much later time $f(n)$. In other words, $\overline{\mathbb{P}}$ learns to assign high probability to the $\overline{\phi}$ faster than $\overline{D}$ can computationally verify them.

> **Analogy: Ramanujan and Hardy.** Imagine that the statements $\overline{\phi}$ are being output by an algorithm that uses heuristics to generate mathematical facts without proofs, playing a role similar to the famously brilliant, often-unrigorous mathematician Srinivasa Ramanujan. Then $\overline{\mathbb{P}}$ plays the historical role of the beliefs of the rigorous G.H. Hardy who tries to verify those results according to a slow deductive process. After Hardy ($\overline{\mathbb{P}}$) verifies enough of Ramanujan's claims ($\phi_{\leq n}$) according to some slow deductive process ($\overline{D}$), he begins to trust Ramanujan, even if the proofs of Ramanujan's later conjectures are impossibly long, putting them ever-further beyond Hardy's current abilities to rigorously verify them. In this story, Hardy's inductive reasoning (and Ramanujan's also) outpaces his deductive reasoning.

This idiom of assigning the right probabilities to ϕ_n no later than day n will be common throughout the paper, so we give it a name.

Definition 4.2.2 (Timely Manner)**.** *Let $\overline{\phi}$ be an e.c. sequence of sentences, and $\overline{p}$ be an e.c. sequence of rational numbers. We say $\overline{\mathbb{P}}$ assigns $\overline{p}$ to $\overline{\phi}$ in a* ***timely manner*** *if for every $\varepsilon > 0$, there exists a time N such that for all $n > N$,*

$$|\mathbb{P}_n(\phi_n) - p_n| < \varepsilon.$$

In other words, $\overline{\mathbb{P}}$ assigns $\overline{p}$ to $\overline{\phi}$ in a timely manner if

$$\mathbb{P}_n(\phi_n) \eqsim_n p_n.$$

Note that there are no requirements on how large N gets as a function of ε. As such, when we say that $\overline{\mathbb{P}}$ assigns $\overline{p}$ to $\overline{\phi}$ in a timely manner, it may take a very long time for convergence to occur. (See the extended paper for a discussion.)

As an example, imagine the reasoner who recognizes that sentences of the form "$1+1+\cdots+1$ is even" are true iff the number of ones is even. Let $\overline{\phi}$ be the sequence where ϕ_n is the version of that sentence with $2n$ ones. If the reasoner starts writing a probability near 100% in the ϕ_n cell by day n at the latest, then intuitively, she has begun incorporating the pattern into her beliefs, and we say that she is assigning high probabilities to $\overline{\phi}$ in a timely manner.

Checking the nth sentence on the nth day is a rather arbitrary choice, and we might hope that a good reasoner would assign high probabilities to e.c. sequences of theorems at a faster rate than that. It is easy to show that this is the case, by the closure properties of efficient computability. In the extended paper, we also show that if $\mathbb{P}_n(\phi_n)$ is close to $\mathbb{P}_\infty(\phi_n)$, then $\mathbb{P}_m(\phi_n)$ is also close to $\mathbb{P}_\infty(\phi_n)$ for $m \geq n$. As such, it doesn't really matter which diagonal $\mathbb{P}_t(\phi_n)$ we check when checking whether or not $\overline{\mathbb{P}}$ has started "recognizing a pattern", so long as t is related to n by a polynomial expression. The most natural choice is to check the probability of the nth sentence on the nth day, and we invite the reader to be on the lookout for expressions of the form "$\mathbb{P}_n(\phi_n)$" as indications that a logical inductor is outpacing its underlying deductive process.

Pushing beyond Theorem 4.2.1, we can show that if $\overline{\mathbb{P}}$ is going to learn to assign $\overline{p}$ to $\overline{\phi}$ at any point in the future, then it learns to assign $\overline{p}$ to $\overline{\phi}$ in a timely manner:

Theorem 4.2.3 (Preemptive Learning)**.** *Let $\overline{\phi}$ be an e.c. sequence of sentences. Then*

$$\liminf_{n\to\infty} \mathbb{P}_n(\phi_n) = \liminf_{n\to\infty} \sup_{m\geq n} \mathbb{P}_m(\phi_n).$$

Furthermore,

$$\limsup_{n\to\infty} \mathbb{P}_n(\phi_n) = \limsup_{n\to\infty} \inf_{m\geq n} \mathbb{P}_m(\phi_n).$$

(Proof in extended paper, B.3.)

For example, consider the sequence

$$\overline{\pi\text{Aeq7}} := \left(\text{“}\pi[A(\underline{n},\underline{n})] = 7\text{”}\right)_{n\in\mathbb{N}^+}$$

where $\pi[i]$ is the ith decimal digit of π and A is the Ackermann function. Each individual sentence is decidable, so the limiting probabilities are 0 for some πAeq7_n and 1 for others. But that pattern of 1s and 0s is not efficiently computable (unless the Ackermann digits of π are easy to predict). Theorem 4.2.3 says that even so, if $\overline{\mathbb{P}}$ is going to learn to assign probability 10% to each πAeq7_n for a long time while it waits to learn the Ackermann numbers, then it learns to assign 10% probability to $\overline{\pi\text{Aeq7}}$ in a timely manner.

This raises the question of whether logical inductors learn to assign probabilities like 10% to sequences like $\overline{\pi\text{Aeq7}}$. We will now answer that question in the affirmative (assuming the Ackermann digits of π are difficult to predict). This follows from the fact that logical inductors learn to use appropriate statistical summaries on sequences of sentences that appear random to $\overline{\mathbb{P}}$. To formalize this claim, we need to formalize the idea that a sequence is “apparently random”. Intuitively, this notion must be defined relative to a specific reasoner and their computational limitations. After all, the digits of π are perfectly deterministic; they only appear random to a reasoner who lacks the resources to compute them. We use the following setup:

Definition 4.2.4 (Divergent Weighting)**.** *A **divergent weighting** $\overline{w} \in [0,1]^{\mathbb{N}^+}$ is an infinite sequence of real numbers in $[0,1]$, such that $\sum_n w_n = \infty$.*

Note that divergent weightings have codomain $[0,1]$ as opposed to $\{0,1\}$, meaning the weightings may single out fuzzy subsets of the sequence. For purposes of intuition, imagine that $\overline{w}$ is a sequence of 0s and 1s, in which case each $\overline{w}$ can be interpreted as a subsequence. The constraint that the w_n sum to ∞ ensures that this subsequence is infinite.

Definition 4.2.5 (Generable From $\overline{\mathbb{P}}$)**.** *A sequence of rational numbers $\overline{q}$ is called **generable from** $\overline{\mathbb{P}}$ if there exists an e.c. $\mathcal{EF}$-progression $\overline{q^\dagger}$ such that $q^\dagger_n(\overline{\mathbb{P}}) = q_n$ for all n. In this case we say that $\overline{q}$ is **$\overline{\mathbb{P}}$-generable**. $\overline{\mathbb{P}}$-generable $\mathbb{R}$-sequences, $\mathbb{Q}$-combination sequences, and $\mathbb{R}$-combination sequences are defined analogously.*

Divergent weightings generable from $\overline{\mathbb{P}}$ are fuzzy subsequences that are allowed to depend continuously (via expressible market features) on the market history. These are, more or less, the pattern-detectors that logical inductors are able to use when searching a sequence for one subsequence that is more likely to be true than all the others. Roughly speaking, if there are no patterns that can be detected in a sequence by $\overline{\mathbb{P}}$-generable divergent weightings, then we say that the sequence is pseudorandom (relative to those weightings):

Definition 4.2.6 (Pseudorandom Sequence)**.** *Given a set S of divergent weightings, a sequence $\overline{\phi}$ of decidable sentences is called **pseudorandom with frequency p** over S if, for all weightings $\overline{w} \in S$,*

$$\lim_{n\to\infty} \frac{\sum_{i\leq n} w_i \cdot \text{Thm}_\Gamma(\phi_i)}{\sum_{i\leq n} w_i}$$

exists and is equal to p.

Note that if the sequence $\overline{\phi}$ is *actually* randomly generated (say, by adding $(c_1, c_2, \ldots)$ to the language of Γ, and tossing a coin weighted with probability $\underline{p}$ towards heads for each i, to determine whether to add c_i or $\neg c_i$ as an axiom) then $\overline{\phi}$ is pseudorandom with frequency p almost surely.[3] Now:

Theorem 4.2.7 (Learning Pseudorandom Frequencies)**.** *If $\overline{\phi}$ be an e.c. sequence of decidable sentences. If $\overline{\phi}$ is pseudorandom over the set of all $\overline{\mathbb{P}}$-generable divergent weightings, then*

$$\mathbb{P}_n(\phi_n) \eqsim_n p.$$

(Proof in extended paper, D.8.)

For example, if the Ackermann digits of n are hard to predict (in the sense that no $\overline{\mathbb{P}}$-generable divergent weighting can single out any fuzzy subset where the digits are particularly likely to be 7), then $\overline{\mathbb{P}}$ assigns probability 10% to $\overline{\pi\text{Aeq}7}$ in a timely manner.

Furthermore, logical inductors learn, in a timely manner, to make their probabilities respect linear inequalities that will hold in the truth values between sentences. For example, consider a computer program `prg` which outputs either 0, 1, or 2 on all inputs, but for which the general case cannot be proven by Γ. Theorem 4.2.1 says that the sequence

$$\overline{\text{prg012}} := \left(\text{“}\underline{\texttt{prg}}(\underline{n}) = 0 \vee \underline{\texttt{prg}}(\underline{n}) = 1 \vee \underline{\texttt{prg}}(\underline{n}) = 2\text{”}\right)_{n \in \mathbb{N}^+}$$

will be learned, in the sense that $\overline{\mathbb{P}}$ will inductively learn to assign each prg012_n a probability near 1 in a timely manner. But what about the following three individual sequences?

$$\begin{aligned}
\overline{\text{prg0}} &:= \left(\text{“}\underline{\texttt{prg}}(\underline{n}) = 0\text{”}\right)_{n \in \mathbb{N}^+} \\
\overline{\text{prg1}} &:= \left(\text{“}\underline{\texttt{prg}}(\underline{n}) = 1\text{”}\right)_{n \in \mathbb{N}^+} \\
\overline{\text{prg2}} &:= \left(\text{“}\underline{\texttt{prg}}(\underline{n}) = 2\text{”}\right)_{n \in \mathbb{N}^+}
\end{aligned}$$

None of the three are purely theorems, so Theorem 4.2.1 does not apply. If they are pseudorandom relative to $\overline{\mathbb{P}}$, then Theorem 4.2.7 says that $\overline{\mathbb{P}}$ will fall back on the limiting frequencies, but that tells us little in cases where there are predictable non-conclusive patterns (e.g., if `prg(i)` is more likely to output 2 when `helper(i)` outputs 17). In fact, the probabilities on the $(\text{prg0}_n, \text{prg1}_n, \text{prg2}_n)$ triplet *should* shift around over time, as $\overline{\mathbb{P}}$ gains new knowledge about related facts and updates its beliefs. How could we tell if those intermediate beliefs were reasonable?

One way is to check their sum. If $\overline{\mathbb{P}}$ believes that `prg(i)` $\in \{0, 1, 2\}$ and it knows how disjunction works, then it should be the case that whenever $\mathbb{P}_n(\text{prg012}_t) \approx 1$, $\mathbb{P}_n(\text{prg0}_t) + \mathbb{P}_n(\text{prg1}_t) + \mathbb{P}_n(\text{prg2}_t) \approx 1$. And this is precisely the case.

Convention 4.2.8 (Constraint)**.** *An $\mathbb{R}$-combination C can be viewed as a* ***constraint****, in which case we say that a valuation $\mathbb{V}$* ***satisfies*** *the constraint if $\mathbb{V}(C) \geq 0$.*

For example, the constraint

$$\text{AND} := -2 + \phi + \psi$$

says that both ϕ and ψ are true, and it is satisfied by $\mathbb{W}$ iff $\mathbb{W}(\phi) = \mathbb{W}(\psi) = 1$. As another example, the pair of constraints

$$\text{XOR} := (1 - \phi - \psi, \phi + \psi - 1)$$

say that exactly one of ϕ and ψ is true, and are satisfied by $\mathbb{P}_7$ iff $\mathbb{P}_7(\phi) + \mathbb{P}_7(\psi) = 1$.

3. Note that actually adding randomness to Γ in this fashion is not allowed, because we assumed that the axioms of Γ are recursively enumerable. It is possible to construct a logical inductor that has access to a source of randomness, by adding one bit of randomness to the market each day, but that topic is beyond the scope of this paper.

Definition 4.2.9 (Bounded Combination Sequences)**.** *By $\mathcal{BCS}(\overline{\mathbb{P}})$ (mnemonic:* ***bounded combination sequences****) we denote the set of all $\overline{\mathbb{P}}$-generable $\mathbb{R}$-combination sequences $\overline{A}$ that are bounded, in the sense that there exists some bound b such that $\|A_n\|_1 \leq b$ for all n, where $\|-\|_1$ includes the trailing coefficient.*

Theorem 4.2.10 (Affine Coherence)**.** *Let $\overline{A} \in \mathcal{BCS}(\overline{\mathbb{P}})$. Then*

$$\liminf_{n\to\infty} \inf_{\mathbb{W}\in\mathcal{C}(\Gamma)} \mathbb{W}(A_n) \leq \liminf_{n\to\infty} \mathbb{P}_\infty(A_n) \leq \liminf_{n\to\infty} \mathbb{P}_n(A_n),$$

and

$$\limsup_{n\to\infty} \mathbb{P}_n(A_n) \leq \limsup_{n\to\infty} \mathbb{P}_\infty(A_n) \leq \limsup_{n\to\infty} \sup_{\mathbb{W}\in\mathcal{C}(\Gamma)} \mathbb{W}(A_n).$$

(Proof in extended paper, C.1.)

These inequalities tie ground truth on $\overline{A}$, to the value of $\overline{A}$ in the limit, to the value of $\overline{A}$ on the main diagonal. In words, it says that if all consistent worlds $\mathbb{W} \in \mathcal{C}(\Gamma)$ value A_n in (a, b) for n large, then $\mathbb{P}_\infty$ values A_n in $(c, d) \subseteq (a, b)$ for n large (because $\mathbb{P}_\infty$ is a weighted mixture of all consistent worlds), and $\overline{\mathbb{P}}$ learns to assign probabilities $\mathbb{P}_n(A_n) \in (c, d)$ in a timely manner. In colloquial terms, $\overline{\mathbb{P}}$ learns in a timely manner to respect *all* linear inequalities that actually hold between sentences, so long as those relationships can be enumerated in polynomial time.

For example, consider the constraint sequence

$$\overline{A} := \left(1 - \mathrm{prg0}_n - \mathrm{prg1}_n - \mathrm{prg2}_n\right)_{n\in\mathbb{N}^+}$$

For all n and all consistent worlds $\mathbb{W} \in \mathcal{C}(\Gamma)$, the value $\mathbb{W}(A_n)$ is 0, so applying Theorem 4.2.10 to $\overline{A}$, we get that $\mathbb{P}_n(A_n) \eqsim_n 0$. By linearity, this means

$$\mathbb{P}_n(\mathrm{prg0}_n) + \mathbb{P}_n(\mathrm{prg1}_n) + \mathbb{P}_n(\mathrm{prg2}_n) \eqsim_n 1,$$

i.e., $\overline{\mathbb{P}}$ learns that the three sequences are mutually exclusive and exhaustive in a timely manner, regardless of how difficult `prg` is to evaluate.

This doesn't mean that $\overline{\mathbb{P}}$ will assign the *correct* values (0 or 1) to each sentence in a timely manner; that would be impossible for a deductively limited reasoner. Rather, $\overline{\mathbb{P}}$'s probabilities will start *satisfying the constraints* in a timely manner. For example, imagine a set of complex constraints holds between seven sequences, such that exactly three sentences in each septuplet are true, but it's difficult to tell which three. Then $\overline{\mathbb{P}}$ will learn this pattern, and start ensuring that its probabilities on each septuplet sum to 3, even if it can't yet assign particularly high probabilities to the correct three.

If we watch an individual septuplet as $\overline{\mathbb{P}}$ reasons, other constraints will push the probabilities on those seven sentences up and down. One sentence might be refuted and have its probability go to zero. Another might get a boost when $\overline{\mathbb{P}}$ discovers that it's likely implied by a high-probability sentence. Another might take a hit when $\overline{\mathbb{P}}$ discovers it likely implies a low-probability sentence. Throughout all this, Theorem 4.2.10 says that $\overline{\mathbb{P}}$ will ensure that the seven probabilities always sum to ≈ 3. $\overline{\mathbb{P}}$'s beliefs on any given day arise from this interplay of many constraints, inductively learned.

4.3 Self-knowledge

Logical inductors also learn to know what they know, and trust their future beliefs, in a way that avoids paradox. For starters,

Theorem 4.3.1 (Introspection)**.** *Let $\overline{\phi}$ be an e.c. sequence of sentences, and $\overline{a}$, $\overline{b}$ be $\overline{\mathbb{P}}$-generable sequences of probabilities. Then, for any e.c. sequence of positive rationals $\overline{\delta} \to 0$, there exists a sequence of positive rationals $\overline{\varepsilon} \to 0$ such that for all n:*

1. *if* $\mathbb{P}_n(\phi_n) \in (a_n + \delta_n, b_n - \delta_n)$*, then*

$$\mathbb{P}_n(\text{“}\underline{a_n} < \underline{\mathbb{P}}_{\underline{n}}(\underline{\phi_n}) < \underline{b_n}\text{”}) > 1 - \varepsilon_n,$$

2. *if* $\mathbb{P}_n(\phi_n) \notin (a_n - \delta_n, b_n + \delta_n)$*, then*

$$\mathbb{P}_n(\text{“}\underline{a_n} < \underline{\mathbb{P}}_{\underline{n}}(\underline{\phi_n}) < \underline{b_n}\text{”}) < \varepsilon_n.$$

(Proof in extended paper, F.1.)

Roughly speaking, this says that if there is an e.c. pattern of the form “your probabilities on $\overline{\phi}$ will be between (a, b)” then $\overline{\mathbb{P}}$ will learn to believe that pattern iff it is true, subject to the caveat that its self-knowledge has only finite precision (i.e., if its beliefs are extremely close to a then it can't always tell which side of a they are on). This “finite-precision self-knowledge” allows logical inductors to avoid the classic paradoxes of self-reference:

Theorem 4.3.2 (Paradox Resistance)**.** *Fix a rational* $p \in (0, 1)$*, and define a sequence of “paradoxical sentences”* $\overline{\chi^p}$ *satisfying*

$$\Gamma \vdash \underline{\chi_n^p} \leftrightarrow \left(\underline{\mathbb{P}}_{\underline{n}}(\underline{\chi_n^p}) < \underline{p}\right)$$

for all n*. Then*

$$\lim_{n\to\infty} \mathbb{P}_n(\chi_n^p) = p.$$

(Proof in extended paper, F.2.)

A logical inductor responds to paradoxical sentences $\overline{\chi^p}$ by assigning probabilities that converge on p. For example, if the sentences say “$\overline{\mathbb{P}}$ will assign me a probability less than 80% on day n”, then $\mathbb{P}_n$ (once $\overline{\mathbb{P}}$ has learned the pattern) starts assigning probabilities extremely close to 80%—so close that polynomial-runtime traders can't tell if it's slightly above or slightly below.

To visualize this, imagine that someone who owns a high-precision brain-scanner and can read off your beliefs, asks you what probability you assign to the claim “you will assign probability <80% to this claim at precisely 10am tomorrow”. As 10am approaches, what happens to your belief in this claim? If you become extremely confident that it's going to be true, then your confidence should drop. But if you become highly confident it's going to be false, then your confidence should spike. Thus, your probabilities should oscillate, pushing your belief so close to 80% that you're not quite sure which way the brain scanner will actually call it. In response to paradoxical sentences, this is exactly how a logical inductor behaves, once it's learned how they work.

To go further, we need to define expected values taken with respect to $\overline{\mathbb{P}}$, which we do as follows.

Definition 4.3.3 (Logically Uncertain Variable)**.** *A* ***logically uncertain variable****, abbreviated* ***LUV****, is any formula* X *free in one variable that defines a unique value via* Γ*, in the sense that*

$$\Gamma \vdash \exists x\colon (X(x) \wedge \forall x'\colon X(x') \to x' = x).$$

Given a LUV X *and a consistent world* $\mathbb{W} \in \mathcal{C}(\Gamma)$*, the* ***value of X in*** $\mathbb{W}$ *is defined to be*

$$\mathbb{W}(X) := \sup_{x\in[0,1]} \mathbb{W}(\text{“}X \geq \underline{x}\text{”}) = 1.$$

In other words, $\mathbb{W}(X)$ *is the supremum of values in* $\mathbb{W}$ *that do not exceed* X*. (This rather roundabout definition is necessary to handle cases where* $\mathbb{W}$ *assigns* X *a non-standard value.)*

We write $\mathcal{U}$ *for the set of all* $[0, 1]$*-LUVs. When manipulating logically uncertain variables, we use shorthand like* “$X < 0.5$” *for* “$\forall x\colon X(x) \to x < 0.5$”*, and so on.*

For example, $H :=$ "$\nu = 0.5$" is a LUV. As another example, $TPC :=$ "$(\nu = 1 \wedge$ Goldbach's conjecture$) \vee (\nu = 0 \wedge \neg$Goldbach's conjecture$)$" is a LUV whose value is somewhat more difficult to determine.

We can now define expectations of LUVs with respect to a given pricing. The first impulse is to use a Riemann sum; this is unsatisfactory, because if $\overline{\mathbb{P}}$ has not yet figured out that a LUV X has a unique value, then it might assign high probability to X being in multiple different places in the $[0,1]$ interval, in which case the expectation of a $[0,1]$-LUV would not necessarily be a $[0,1]$-LUV. So instead, we define expectations using an analog of a cumulative density function:

Definition 4.3.4 (Expectation)**.** *For a given valuation $\mathbb{V}$, we define the **approximate expectation operator** $\mathbb{E}_k^{\mathbb{V}}$ for $\mathbb{V}$ with precision k by*

$$\mathbb{E}_k^{\mathbb{V}}(X) := \sum_{i=0}^{k-1} \frac{1}{k} \mathbb{V}(\text{“}\underline{X} > \underline{i}/\underline{k}\text{”}).$$

where X is a $[0,1]$-LUV.

We will often want to take a limit of $\mathbb{E}_k^{\mathbb{P}_n}(X)$ as both k and n approach ∞. We hereby make the fairly arbitrary choice to focus on the case $k = n$ for simplicity, adopting the shorthand

$$\mathbb{E}_n := \mathbb{E}_n^{\mathbb{P}_n}$$

In other words, when we examine how a logical inductor's expectations change on a sequence of sentences over time, we will (arbitrarily) consider approximate expectations that gain in precision at a rate of one unit per day.

We will now show that the expectation operator $\mathbb{E}_n$ possesses properties that make it worthy of that name.

Theorem 4.3.5 (Expectations Converge)**.** *The limit $\mathbb{E}_\infty : \mathcal{S} \to [0,1]$ defined by*

$$\mathbb{E}_\infty(X) := \lim_{n\to\infty} \mathbb{E}_n(X)$$

exists for all $X \in \mathcal{U}$. (Proof in extended paper, E.4.)

Theorem 4.3.6 (Linearity of Expectation)**.** *Let $\overline{\alpha}, \overline{\beta}$ be bounded e.c. sequences of rational numbers expressible from $\overline{\mathbb{P}}$, and let $\overline{X}, \overline{Y}$, and $\overline{Z}$ be e.c. sequences of $[0,1]$-LUVs. If we have $\Gamma \vdash Z_n = \alpha_n X_n + \beta_n Y_n$ for all n, then*

$$\alpha_n \mathbb{E}_n(X_n) + \beta_n \mathbb{E}_n(Y_n) \eqsim_n \mathbb{E}_n(Z_n).$$

(Proof in extended paper, E.9.)

Theorem 4.3.7 (Expectation Coherence)**.** *Let $\mathcal{BLCS}(\overline{\mathbb{P}})$ be the set of all bounded LUV-combination sequences, defined analogously to definitions 3.4.6, 4.2.5, and 4.2.9. (See the appendix for details.) Let $\overline{B} \in \mathcal{BLCS}(\overline{\mathbb{P}})$. Then*

$$\liminf_{n\to\infty} \inf_{\mathbb{W}\in\mathcal{C}(\Gamma)} \mathbb{W}(B_n) \le \liminf_{n\to\infty} \mathbb{E}_\infty(B_n) \le \liminf_{n\to\infty} \mathbb{E}_n(B_n),$$

and

$$\limsup_{n\to\infty} \mathbb{E}_n(B_n) \le \limsup_{n\to\infty} \mathbb{E}_\infty(B_n) \le \limsup_{n\to\infty} \sup_{\mathbb{W}\in\mathcal{C}(\Gamma)} \mathbb{W}(B_n).$$

(Proof in extended paper, E.7.)

Many other properties of the expectation operator are discussed in the extended version.

We can now show that logical inductors trust their future beliefs:

Definition 4.3.8 (Deferral Function)**.** *A function $f : \mathbb{N}^+ \to \mathbb{N}^+$ is called a **deferral function** if*

1. *$f(n) > n$ for all n, and*

2. *$f(n)$ can be computed in time polynomial in $f(n)$, i.e., if there is some algorithm and a polynomial function h such that for all n, the algorithm computes $f(n)$ within $h(f(n))$ steps.*

*If f is a deferral function, we say that f **defers** n to $f(n)$.*

Theorem 4.3.9 (No Expected Net Update)**.** *Let f be a deferral function, and let $\overline{\phi}$ be an e.c. sequence of sentences. Then*

$$\mathbb{P}_n(\phi_n) \eqsim_n \mathbb{E}_n(\text{“}\underline{\mathbb{P}}_{\underline{f(n)}}(\underline{\phi_n})\text{”}).$$

(Proof in extended paper, F.6.)

Roughly speaking, this says that if $\overline{\mathbb{P}}$ on day n believes that on day $f(n)$ it will believe ϕ_n with high probability, then it already believes ϕ_n with high probability today. In other words, logical inductors learn to adopt their predicted future beliefs as their current beliefs in a timely manner—they don't say "tomorrow I expect to believe that ϕ is true, but today I think it's false".

We will also show that, roughly speaking, logical inductors trust that if their beliefs change, then they must have changed for good reasons. To do this, we first define an indicator LUV $\mathbb{1}(\phi)$, which is 1 if ϕ is true and 0 otherwise, and a continuous threshold indicator $\mathrm{Ind}_\delta(x > y)$ which is 1 if x is unambiguously greater than y, 0 if x is less than y, and intermediate when the case is unclear.

Definition 4.3.10 (Indicator LUVs)**.** *For any sentence ϕ, we define its **indicator LUV** by the formula*

$$\mathbb{1}(\phi) := \text{“}(\underline{\phi} \wedge (\nu = 1)) \vee (\neg\underline{\phi} \wedge (\nu = 0))\text{”}.$$

Definition 4.3.11 (Continuous Threshold Indicator)**.** *Let $\delta > 0$ be a rational number, and x and y be real numbers. We then define*

$$\mathrm{Ind}_\delta(x > y) := \begin{cases} 0 & \text{if } x \leq y \\ \dfrac{x-y}{\delta} & \text{if } \quad y < x \leq y + \delta \\ 1 & \text{if } \quad\quad y + \delta < x. \end{cases}$$

Notice that $\mathrm{Ind}_\delta(x > y)$ has no false positives, and that it is linear in the region between y and $y + \delta$. We define $\mathrm{Ind}_\delta(x < y)$ analogously, and we define

$$\mathrm{Ind}_\delta(a < x < b) := \min(\mathrm{Ind}_\delta(x > a), \mathrm{Ind}_\delta(x < b)).$$

Observe that we can generalize this definition to the case where x and y are expressible features, in which case $\mathrm{Ind}_\delta(x > y)$ is an expressible $[0, 1]$-feature.

Then:

Theorem 4.3.12 (Self-Trust)**.** *Let f be a deferral function, $\overline{\phi}$ be an e.c. sequence of sentences, $\overline{\delta}$ be an e.c. sequence of positive rational numbers, and $\overline{p}$ be a $\overline{\mathbb{P}}$-generable sequence of probabilities. Then*

$$\mathbb{E}_n\left(\text{“}\underline{\mathbb{1}(\phi_n)} \cdot \underline{\mathrm{Ind}_{\delta_n}}\left(\underline{\mathbb{P}}_{\underline{f(n)}}(\underline{\phi_n}) > \underline{p_n}\right)\text{”}\right) \gtrsim_n p_n \cdot \mathbb{E}_n\left(\text{“}\underline{\mathrm{Ind}_{\delta_n}}\left(\underline{\mathbb{P}}_{\underline{f(n)}}(\underline{\phi_n}) > \underline{p_n}\right)\text{”}\right).$$

(Proof in extended paper, F.8.)

Very roughly speaking, if we squint at Theorem 4.3.12, it says something like

$$\mathbb{E}_{\text{now}}(\phi \mid P_{\text{later}}(\phi) > p) \gtrsim p,$$

i.e., if we ask $\overline{\mathbb{P}}$ what it would believe about ϕ now if it learned that it was going to believe ϕ with probability at least p in the future, then it will answer with a probability that is at least p.

As a matter of fact, Theorem 4.3.12 actually says something slightly weaker, which is also more desirable. Let each ϕ_n be the self-referential sentence "$\underline{\mathbb{P}}_{f(\underline{n})}(\underline{\phi_n}) < 0.5$" which says that the future $\mathbb{P}_{f(n)}$ will assign probability less than 0.5 to ϕ_n. Then, conditional on $\mathbb{P}_{f(n)}(\phi_n) \geq 0.5$, $\mathbb{P}_n$ should believe that the probability of ϕ_n is 0. And indeed, this is what a logical inductor will do:

$$\mathbb{P}_n\left(\text{“}\underline{\phi_n} \wedge (\underline{\mathbb{P}}_{f(\underline{n})}(\underline{\phi_n}) \geq 0.5)\text{”}\right) \eqsim_n 0,$$

because each of those conjunctions is disprovable. This is why Theorem 4.3.12 uses continuous indicator functions: With discrete conjunctions, the result would be undesirable (not to mention false).

What Theorem 4.3.12 says is that $\overline{\mathbb{P}}$ attains self trust of the "if in the future I will believe x is very likely, then it must be because x is very likely" variety, given finite-precision access to its future beliefs. In doing so, it retains the ability to think it can outperform its future self's beliefs when its future self confronts paradoxes. In colloquial terms, if we ask "what's your probability on the paradoxical sentence ϕ_n given that your future self believes it with probability *exactly* 0.5?" then $\overline{\mathbb{P}}$ will answer "very low", but if we ask "what's your probability on the paradoxical sentence ϕ_n given that your future self believes it with probability *extremely close to* 0.5?" then $\overline{\mathbb{P}}$ will answer "roughly 0.5."

Still speaking roughly, this means that logical inductors trust their future beliefs to be accurate and only change for good reasons. Theorem 4.3.12 says that if you ask "what's the probability of ϕ, given that in the future you're going to believe it's more than 95% likely?" then you'll get an answer that's no less than 0.95, even if the logical inductor currently thinks that ϕ is unlikely.

5 Discussion

In the extended version of this paper, we give a logical induction algorithm, and provide proofs of the above theorems, and discuss a number of other types of properties (relating, e.g., to calibration and conditional probabilities). We also defer an extended discussion of this framework and its applications to the extended version.

5.1 Acknowledgements

We acknowledge Abram Demski, Benya Fallenstein, Daniel Filan, Eliezer Yudkowsky, Jan Leike, János Kramár, Nisan Stiennon, Patrick LaVictoire, Paul Christiano, Sam Eisenstat, Scott Aaronson, and Vadim Kosoy, for valuable comments and discussions. We also acknowledge contributions from attendees of the MIRI summer fellows program, the MIRIxLA group, and the MIRIχ group.

This research was supported as part of the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant #2015-144576.

References

Aaronson, Scott. 2013. "Why philosophers should care about computational complexity." In *Computability: Turing, Gödel, Church, and Beyond,* edited by B. Jack Copeland, Carl J. Posy, and Oron Shagrir. MIT Press.

Beygelzimer, Alina, John Langford, and David M. Pennock. 2012. "Learning Performance of Prediction Markets with Kelly Bettors." In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012),* edited by Vincent Conitzer, Michael Winikoff, Wiebe van der Hoek, and Lin Padgham, 1317–1318. International Foundation for Autonomous Agents / Multiagent Systems.

Boole, George. 1854. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities.* Dover Publications.

Briol, François-Xavier, Chris Oates, Mark Girolami, Michael A. Osborne, and Dino Sejdinovic. 2015. "Probabilistic Integration." arXiv: `1512.00933 [stat.ML]`.

Christiano, Paul. 2014. *Non-Omniscience, Probabilistic Inference, and Metamathematics.* Technical report 2014–3. Berkeley, CA: Machine Intelligence Research Institute. `http://intelligence.org/files/Non-Omniscience.pdf`.

de Finetti, Bruno. 1937. "Foresight: Its Logical Laws, Its Subjective Sources." In *Studies in Subjective Probability,* edited by Henry E. Kyburg and Howard E.K. Smokler. Huntington, New York: Roger E. Kreiger Publishing Co.

Demski, Abram. 2012. "Logical Prior Probability." Edited by Joscha Bach, Ben Goertzel, and Matthew Iklé. *Artificial General Intelligence. 5th International Conference, AGI 2012* (New York), no. 7716: 50–59.

Gaifman, Haim. 1964. "Concerning Measures in First Order Calculi." *Israel Journal of Mathematics* 2 (1): 1–18.

Garrabrant, Scott, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and Jessica Taylor. 2016. "Logical Induction." `https://intelligence.org/files/LogicalInduction.pdf`.

Good, Irving J. 1950. *Probability and the Weighing of Evidence.* Charles Griffin, London.

Hacking, Ian. 1967. "Slightly More Realistic Personal Probability." *Philosophy of Science* 34 (4): 311–325.

Hutter, Marcus, John W. Lloyd, Kee Siong Ng, and William T. B. Uther. 2013. "Probabilities on Sentences in an Expressive Logic." *Journal of Applied Logic* 11 (4): 386–420.

Li, Ming, and Paul M. B. Vitányi. 1993. *An Introduction to Kolmogorov Complexity and its Applications.* 1st ed. New York: Springer.

Potyka, Nico, and Matthias Thimm. 2015. "Probabilistic Reasoning with Inconsistent Beliefs Using Inconsistency Measures." In *24th International Joint Conference on Artificial Intelligence (IJCAI-15),* 3156–3163. Buenos Aires, Argentina: AAAI Press.

Ramsey, Frank Plumpton. 1931. "Truth and Probability." In *The Foundations of Mathematics and other Logical Essays,* edited by Richard Bevan Braithwaite, 156–198. New York: Harcourt, Brace.

Richardson, Matthew, and Pedro Domingos. 2006. "Markov Logic Networks." *Machine Learning* 62 (1-2): 107–136.

Solomonoff, Ray J. 1964. "A Formal Theory of Inductive Inference. Part I." *Information and Control* 7 (1): 1–22.

von Neumann, John, and Oskar Morgenstern. 1944. *Theory of Games and Economic Behavior.* 1st ed. Princeton, NJ: Princeton University Press.

Zhang, Yitang. 2014. "Bounded gaps between primes." *Annals of Mathematics* 179 (3): 1121–1174.

Zvonkin, Alexander K., and Leonid A. Levin. 1970. "The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms." *Russian Mathematical Surveys* 25 (6): 83–124.